Trabajo Practico N° 1 - 75.06 Organizacion de Datos - Grupo 28

Analisis exploratorio de un dataset con informacion provista por Navent

Link al repositorio de GitHub: https://github.com/Andivisciglio/TP1-OrganizacionDeDatos-Grupo28

Comienza el analisis.

```
In [1]: import datetime as datetime
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

        %matplotlib inline

        plt.style.use('default')

        sns.set(style="whitegrid")
```

```
In [2]: postulantes_edu_df = pd.read_csv('fiuba_1_postulantes_educacion.csv')
        postulantes_gye_df = pd.read_csv('fiuba_2_postulantes_genero_y_edad.csv')
        vistas_df = pd.read_csv('fiuba_3_vistas.csv')
        postulaciones_df = pd.read_csv('fiuba_4_postulaciones.csv')
        avisos_online_df = pd.read_csv('fiuba_5_avisos_online.csv')
        avisos_detalle_df = pd.read_csv('fiuba_6_avisos_detalle.csv')
```

3 Analizamos la informacion disponible

```
In [3]: postulaciones_df.columns
```

```
Out[3]: Index(['idaviso', 'idpostulante', 'fechapostulacion'], dtype='object')
```

```
In [4]: #Verificamos que no haya nulos en las columnas del dataframe
        print(postulaciones_df['idaviso'].isnull().any())
        print(postulaciones_df['fechapostulacion'].isnull().any())
        print(postulaciones_df['idpostulante'].isnull().any())
```

```
False
False
False
```

```
In [5]: avisos_detalle_df.columns
```

```
Out[5]: Index(['idaviso', 'idpais', 'titulo', 'descripcion', 'nombre_zona', 'ciudad',
               'mapacalle', 'tipo_de_trabajo', 'nivel_laboral', 'nombre_area',
               'denominacion_empresa'],
              dtype='object')
```

```
In [6]: print(avisos_detalle_df['idaviso'].isnull().any())
        print(avisos_detalle_df['idpais'].isnull().any())
        print(avisos_detalle_df['titulo'].isnull().any())
        print(avisos_detalle_df['mapacalle'].isnull().any())
        print(avisos_detalle_df['tipo_de_trabajo'].isnull().any())
        print(avisos_detalle_df['nivel_laboral'].isnull().any())
        print(avisos_detalle_df['nombre_area'].isnull().any())
        print(avisos_detalle_df['descripcion'].isnull().any())
        print(avisos_detalle_df['titulo'].isnull().any())
        print(avisos_detalle_df['idaviso'].isnull().any())
        print(avisos_detalle_df['ciudad'].isnull().any())
        print(avisos_detalle_df['nombre_zona'].isnull().any())
```

```
False
False
False
True
False
False
False
False
False
False
True
False
```

```
In [7]: #Observo que solo 47 avisos de 13mil tienen ciudad especificada.
        #No es suficiente como para generalizar conclusiones.
        #Eliminamos la columna "ciudad"
        print(avisos_detalle_df['ciudad'].shape)
        print(avisos_detalle_df['ciudad'].count())
        avisos_detalle_df = avisos_detalle_df.drop('ciudad', 1)
```

```
(13534,)
47
```

```
In [8]: #Observamos que la columna "idpais" siempre tiene el mismo valor.
        print(avisos_detalle_df['idpais'].value_counts())
```

```
1    13534
Name: idpais, dtype: int64
```

```
In [9]: #Como no brinda informacion relevante, la eliminamos.
        avisos_detalle_df = avisos_detalle_df.drop('idpais',1)
```

```
In [10]: #Observamos que la informacion utilizable hace referencia a CABA vs GBA, por lo tanto
         print(avisos_detalle_df['nombre_zona'].value_counts())
         avisos_detalle_df = avisos_detalle_df[ (avisos_detalle_df['nombre_zona'] == 'GBA Oest
         avisos_detalle_df = avisos_detalle_df[ (avisos_detalle_df['nombre_zona'] == 'Buenos Ai
         print()
         print(avisos_detalle_df['nombre_zona'].value_counts())
```

```
Gran Buenos Aires              12654
Capital Federal                  876
Buenos Aires (fuera de GBA)        2
GBA Oeste                          2
Name: nombre_zona, dtype: int64

Gran Buenos Aires    12654
Capital Federal        876
Name: nombre_zona, dtype: int64
```

```
In [11]: #Vemos que los datos del archivo de vistas son limitados a unos pocos dias del mes de
         vistas_df['timestamp'] = pd.to_datetime(vistas_df['timestamp'])
         print(vistas_df['timestamp'].dt.day.value_counts())
         print(vistas_df['timestamp'].dt.month.value_counts())
```

```
27    232145
26    227957
28    227160
24     95930
25     90646
23     47236
1      40823
Name: timestamp, dtype: int64
2    921074
3     40823
Name: timestamp, dtype: int64
```

```
In [12]: #Observamos que tanto para el dia 23 como para el dia 1, hay muy poca informacion.
         df = vistas_df[vistas_df['timestamp'].dt.day == 1 ]
         print(df['timestamp'].dt.hour.value_counts())
         df = vistas_df[vistas_df['timestamp'].dt.day == 23 ]
         print(df['timestamp'].dt.hour.value_counts())
```

```
0    9830
1    9810
2    9283
3    7163
4    4737
Name: timestamp, dtype: int64
19    10370
20     9644
21     9081
22     7181
23     7041
18     3919
Name: timestamp, dtype: int64
```

```
In [13]: #Al trabajar con los datos de los postulantes, se ve que hay fechas nulas.
         print(postulantes_gye_df.isnull().any())
         print()
         print(postulantes_gye_df['fechanacimiento'].isnull().value_counts())
         print()
         print('True indica la cantidad de fechas nulas.')
```

```
idpostulante       False
fechanacimiento     True
sexo               False
dtype: bool

False    196138
True       4750
Name: fechanacimiento, dtype: int64

True indica la cantidad de fechas nulas.
```

```
In [14]: postulantes_gye_df['sexo'].value_counts()
```

```
Out[14]: FEM           101981
         MASC           94339
         NO_DECLARA      4568
         Name: sexo, dtype: int64
```

```
In [15]: #Se ve que la mayoria de entradas con fecha nula, estan asociadas a sexo no declarado
         postulantes_gye_df = postulantes_gye_df.dropna()
         postulantes_gye_df['sexo'].value_counts()
```

```
Out[15]: FEM           101677
         MASC           94016
         NO_DECLARA       445
         Name: sexo, dtype: int64
```

In [16]:
```
#Como la cantidad de entradas con sexo no declarado bajo considerablemente, no es suf
#Eliminamos esas entradas
postulantes_gye_df = postulantes_gye_df[ (postulantes_gye_df['sexo'] == 'NO_DECLARA')
```

In [17]:
```
postulantes_gye_df['sexo'].value_counts()
```

Out[17]:
```
FEM     101677
MASC     94016
Name: sexo, dtype: int64
```

In [18]:
```
#Tambien en este archivo se puede ver que hay fechas anormales, que seran filtradas l
print('Entradas con fechas incorrectas :')
print(postulantes_gye_df[postulantes_gye_df['fechanacimiento'].str.startswith('00')])
```

```
Entradas con fechas incorrectas :
       idpostulante fechanacimiento  sexo
56206       xkPwXwY      0031-12-11   FEM
71458       LN85Y3b      0029-05-11  MASC
130846      8M2R6pz      0024-02-09   FEM
141832      A36Npjj      0033-09-14   FEM
148638      GNZOvAv      0004-07-19  MASC
149653      1QPQ8QL      0011-03-08  MASC
```

In [19]:
```
#Analizando la cantidad de postulaciones por usuario, nos encontramos con tres casos
#de postulaciones por usuario.
#Sin embargo los datos son irrelevantes a la hora de analizar el archivo en su totali
print(postulaciones_df['idpostulante'].value_counts().head(5))
print()
print('El promedio de postulaciones por usuario es: ', postulaciones_df['idpostulante
```

```
axmjOE     3166
6rQdqjl    1773
5Mwjak     1485
8MaQjP3    1414
4rNzLje    1213
Name: idpostulante, dtype: int64

El promedio de postulaciones por usuario es:  16.93293277846362
```

In [20]:
```
#A la hora de analizar la descripcion de los avisos, se puede ver que varios de ellos
#Esta informacion relevante no figura en ninguna columna del set de datos.


df = avisos_detalle_df[avisos_detalle_df['descripcion'].str.contains('Cordoba')]
df = df['descripcion'].value_counts().to_frame().reset_index()
df.iloc[2,0]
```

Out[20]:
```
'<p>En\xa0<strong>Garbarino</strong>\xa0nos encontramos en la búsqueda de nuevos tale
```

In [21]:
```
#Otro ejemplo con mendoza.
df = avisos_detalle_df[avisos_detalle_df['descripcion'].str.contains('Mendoza')]
df = df['descripcion'].value_counts().to_frame().reset_index()
df.iloc[5,0]
```

Out[21]:
```
'<p style=""><span style="">MANPOWER seleccionará</span> <span style="">un Consultor
```

In [22]:
```
#La cantidad de informacion sobre mapacalle coincide con la cantidad de avisos en cap

print(avisos_detalle_df[avisos_detalle_df['nombre_zona'] == 'Capital Federal'].shape)
print()
print(avisos_detalle_df['mapacalle'].count())
```

```
(876, 9)

871
```

In [23]:
```
#Filtro para avisos en capital federal y observo que el mapacalle esta solamente asoc

print(avisos_detalle_df['mapacalle'].count())
avisos_caba_df = avisos_detalle_df[avisos_detalle_df['nombre_zona'] == 'Capital Federa
print(avisos_caba_df.shape)
print(avisos_caba_df['mapacalle'].count())
```

```
871
(876, 9)
871
```

In [24]:
```
avisos_caba_df.head()
```

Out[24]:

	idaviso	titulo	descripcion	nombre_zona	mapacalle	tipo_de_trabajo	nivel_laboral
2	1000150677	Chofer de taxi	<p>TE GUSTA MANEJAR? QUERES GANAR PLATA HACIEN...	Capital Federal	Empedrado 2336	Full-time	Senior / Semi-Senior
8	9240880	Productores Asesores Independiente, para venta...	Agente\r\noficial Selecciona:<...	Capital Federal	NaN	Full-time	Jefe / Supervisor / Responsable
24	1110640622	Agente Inmobiliario para Oficinas RE/MAX en Zo...	<p>Si te gusta deleitar a tus clientes, brinda...	Capital Federal			
34	1111035451	Checker de procesos (laboratorio)	<p>Importante laboratorio farmacéutico naciona...	Capital Federal			
46	1111174081	Gestor de Cobranzas - Telecobrador	<p>En MAS ACTIVOS BPO te estamos esper...	Capital Federal			

```
24  Gobernador Irigoyen 56       Full-time            Senior / Semi-Senior
34         Cochabamba 2525       Full-time            Senior / Semi-Senior
46                 ALEM 116      Part-time            Senior / Semi-Senior

     nombre_area              denominacion_empresa
2     Transporte                        FAMITAX SRL
8      Comercial  Agencia Oficial Alejandro Arizaga
24     Comercial                  RE/MAX TITANIUM
34  Farmacéutica                GI GROUP Argentina
46   Call Center                  MAS ACTIVOS S.A.
```

4 £Que nivel laboral tiene mayor oferta y demanda?

```
In [25]: #Comenzamos uniendo los datos de los archivos de avisos y postulaciones.
         merge_detalle_postulaciones_df = postulaciones_df.merge(avisos_detalle_df, on = 'idav
```

```
In [26]: #Confirmamos que la union se hizo correctamente
         print(merge_detalle_postulaciones_df.count())
         merge_detalle_postulaciones_df.head()
```

```
idaviso                3073975
idpostulante           3073975
fechapostulacion       3073975
titulo                 3073975
descripcion            3073975
nombre_zona            3073975
mapacalle               205266
tipo_de_trabajo        3073975
nivel_laboral          3073975
nombre_area            3073975
denominacion_empresa   3073725
dtype: int64
```

```
Out[26]:       idaviso idpostulante     fechapostulacion  \
         0  1112257047         NM5M  2018-01-15 16:22:34
         1  1112257047       1kJqGb  2018-01-15 10:23:11
         2  1112257047       eOE9Rr  2018-01-15 10:42:07
         3  1112257047       Zrx8Xz  2018-01-30 10:18:14
         4  1112257047       ZrKNQY  2018-01-16 17:19:38

                                                titulo  \
         0  REPRESENTANTES DE ATENCIÓN AL CLIENTE/ RETENCIÓN
         1  REPRESENTANTES DE ATENCIÓN AL CLIENTE/ RETENCIÓN
         2  REPRESENTANTES DE ATENCIÓN AL CLIENTE/ RETENCIÓN
         3  REPRESENTANTES DE ATENCIÓN AL CLIENTE/ RETENCIÓN
         4  REPRESENTANTES DE ATENCIÓN AL CLIENTE/ RETENCIÓN
```

```
                                          descripcion        nombre_zona  \
0  <p><strong><em><span style="">En </span></em><...  Gran Buenos Aires
1  <p><strong><em><span style="">En </span></em><...  Gran Buenos Aires
2  <p><strong><em><span style="">En </span></em><...  Gran Buenos Aires
3  <p><strong><em><span style="">En </span></em><...  Gran Buenos Aires
4  <p><strong><em><span style="">En </span></em><...  Gran Buenos Aires

  mapacalle tipo_de_trabajo nivel_laboral          nombre_area  \
0       NaN       Part-time          Otro  Atención al Cliente
1       NaN       Part-time          Otro  Atención al Cliente
2       NaN       Part-time          Otro  Atención al Cliente
3       NaN       Part-time          Otro  Atención al Cliente
4       NaN       Part-time          Otro  Atención al Cliente

     denominacion_empresa
0  Eficasia Argentina SA
1  Eficasia Argentina SA
2  Eficasia Argentina SA
3  Eficasia Argentina SA
4  Eficasia Argentina SA
```

In [27]:
```
#Se identifican las variables categoricas de la columna "nivel Laboral"
avisos_detalle_df['nivel_laboral'].unique()
```

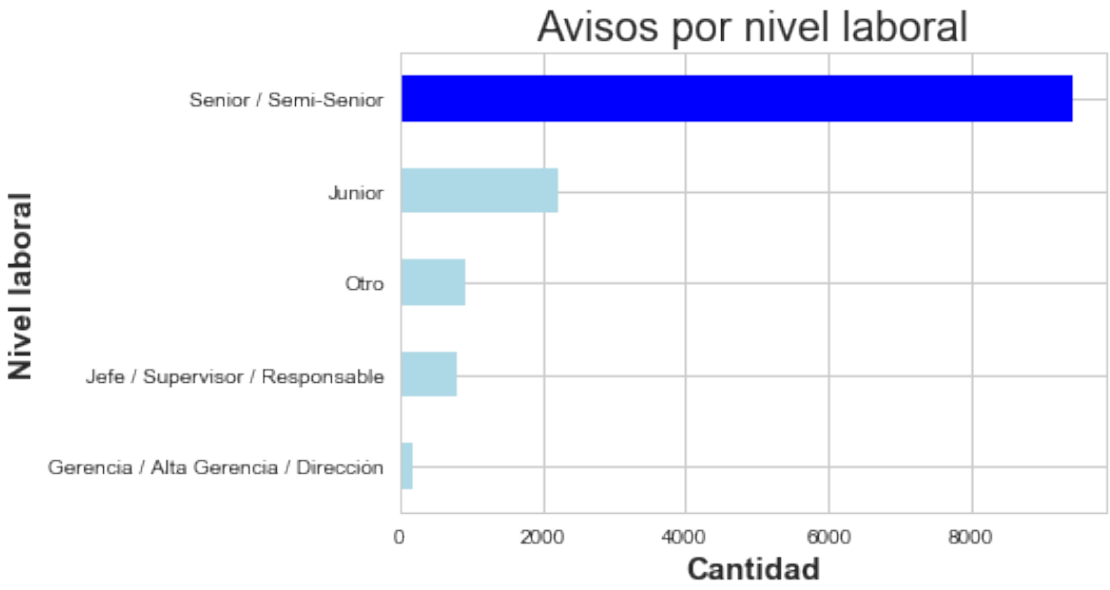
Out[27]:
```
array(['Senior / Semi-Senior', 'Junior',
       'Jefe / Supervisor / Responsable', 'Otro',
       'Gerencia / Alta Gerencia / Dirección'], dtype=object)
```

In [28]:
```
avisos_detalle_df['nivel_laboral'].value_counts()
```

Out[28]:
```
Senior / Semi-Senior                   9406
Junior                                 2215
Otro                                    919
Jefe / Supervisor / Responsable         809
Gerencia / Alta Gerencia / Dirección     181
Name: nivel_laboral, dtype: int64
```

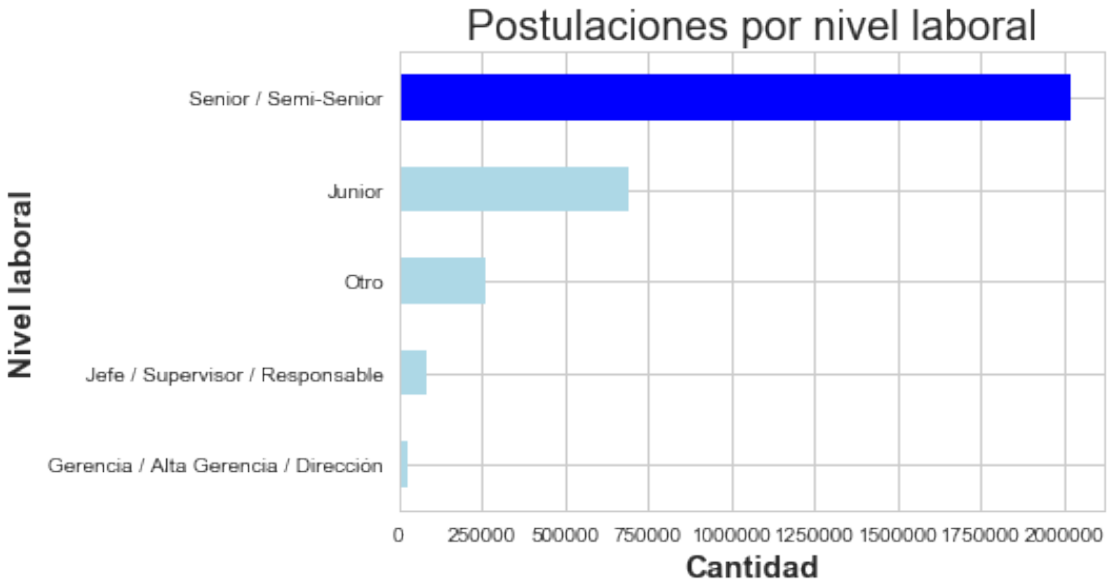
In [29]:
```
#Ahora graficamos la cantidad de avisos para cada nivel laboral, usando solo el archi
avisos_por_nivel = avisos_detalle_df['nivel_laboral'].value_counts(ascending = True).p
avisos_por_nivel.set_title("Avisos por nivel laboral", fontsize = 20)
avisos_por_nivel.set_xlabel("Cantidad", fontsize = 15, weight = 'bold')
avisos_por_nivel.set_ylabel("Nivel laboral", fontsize = 15, weight = 'bold')
```

Out[29]:
```
Text(0,0.5,'Nivel laboral')
```



In [30]:
```
#Ahora lo mismo que antes, pero con el data frame que une los datos.
postulaciones_por_nivel = merge_detalle_postulaciones_df['nivel_laboral'].value_counts
postulaciones_por_nivel.set_title("Postulaciones por nivel laboral", fontsize = 20)
postulaciones_por_nivel.set_xlabel("Cantidad", fontsize = 15, weight = 'bold')
postulaciones_por_nivel.set_ylabel("Nivel laboral", fontsize = 15, weight = 'bold')
```

Out[30]:
```
Text(0,0.5,'Nivel laboral')
```



Como se puede ver, ambos gráficos son de aspecto similar, por lo que se mantiene la relación de oferta y demanda para los diferentes niveles laborales. Claramente el nivel mas apuntado por las empresas a la hora de buscar postulantes es Senior / Semi-Senior, y a su vez es el nivel con mayor demanda, lo que puede indicar que es el sector mas activo del mundo laboral.

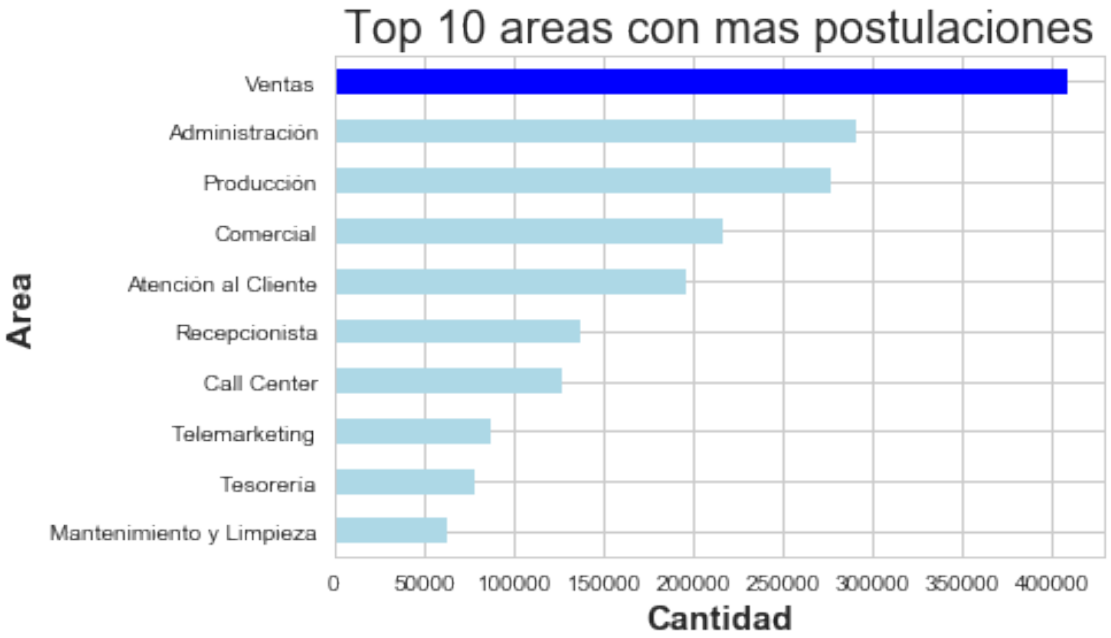
5 £Como es la relacion entre oferta y demanda en las diferentes areas laborales?

In [31]:
```
#Comenzamos con un grafico que muestra el Top 10 de areas con mas avisos.
avisos_por_nombre_area = avisos_detalle_df['nombre_area'].value_counts().head(10).sor
avisos_por_nombre_area.set_title("Top 10 areas con mas avisos", fontsize = 20)
avisos_por_nombre_area.set_xlabel("Cantidad",fontsize = 15, weight = 'bold')
avisos_por_nombre_area.set_ylabel("Area", fontsize = 15, weight = 'bold')
```

Out[31]: Text(0,0.5,'Area')

In [32]:
```
#Ahora lo mismo pero para las potulaciones.
postulaciones_por_nombre_area = merge_detalle_postulaciones_df['nombre_area'].value_c
postulaciones_por_nombre_area.set_title("Top 10 areas con mas postulaciones", fontsiz
postulaciones_por_nombre_area.set_xlabel("Cantidad",fontsize = 15, weight = 'bold')
postulaciones_por_nombre_area.set_ylabel("Area", fontsize = 15, weight = 'bold')
```

Out[32]: Text(0,0.5,'Area')



Empezamos a relacionar la cantidad de postulaciones con la cantidad de avisos. La condicion esta dada por el 25% del promedio de avisos aproximadamente.

```
In [33]: merge = avisos_detalle_df['nombre_area'].value_counts().to_frame().reset_index()
         columnas = ['nombre_area','cantidad_avisos']
         merge.columns = columnas
         merge = merge[(merge['cantidad_avisos']) > 30]
         merge.head()
```

Out[33]:

	nombre_area	cantidad_avisos
0	Ventas	1658
1	Comercial	983
2	Administración	901
3	Producción	821
4	Programación	576

```
In [34]: df = merge_detalle_postulaciones_df['nombre_area'].value_counts().to_frame().reset_in
         columnas = ['nombre_area','cantidad_postulaciones']
         df.columns = columnas

         df.head()
```

Out[34]:

	nombre_area	cantidad_postulaciones
0	Ventas	408148
1	Administración	291135
2	Producción	277089
3	Comercial	216677
4	Atención al Cliente	195636

In [35]:
```
merge = merge.merge(df, on = 'nombre_area')
merge.head()
```

Out[35]:

	nombre_area	cantidad_avisos	cantidad_postulaciones
0	Ventas	1658	408148
1	Comercial	983	216677
2	Administración	901	291135
3	Producción	821	277089
4	Programación	576	8188

In [36]:
```
#Obtenemos la relacion postulaciones/avisos.
merge['postulaciones/avisos'] = ( merge['cantidad_postulaciones'] ) / (merge['cantidad

merge.head()
```

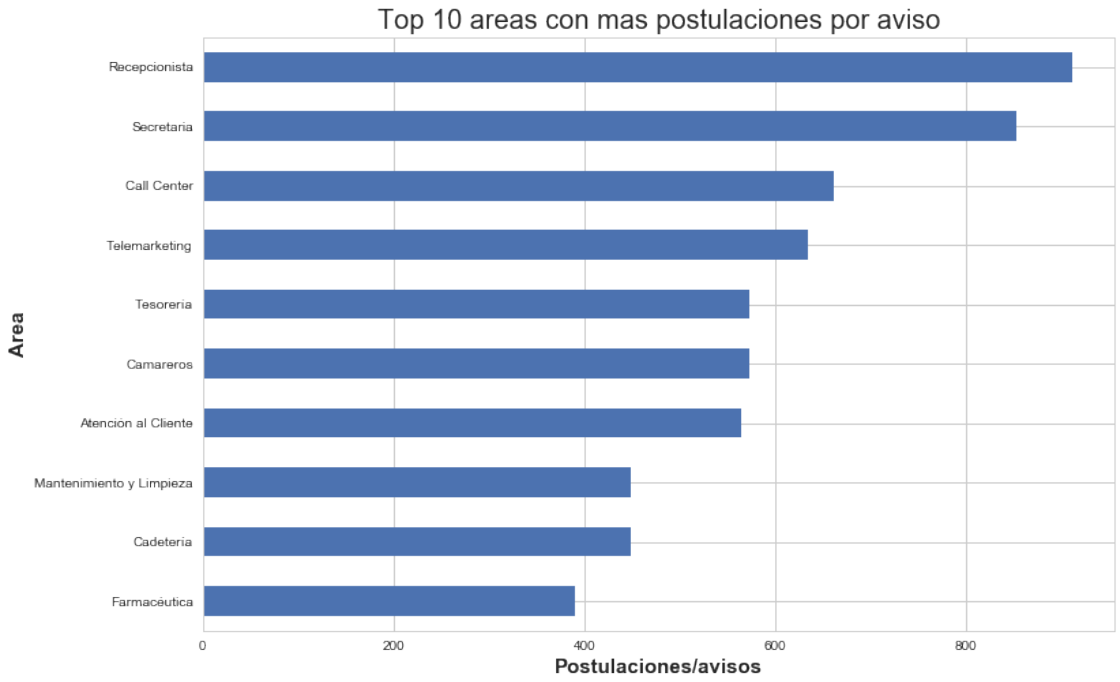
Out[36]:

	nombre_area	cantidad_avisos	cantidad_postulaciones	\
0	Ventas	1658	408148	
1	Comercial	983	216677	
2	Administración	901	291135	
3	Producción	821	277089	
4	Programación	576	8188	

	postulaciones/avisos
0	246.168878
1	220.424212
2	323.124306
3	337.501827
4	14.215278

In [37]:
```
#Grafico de relacion entre cantidad de postulaciones y cantidad de avisos por nombre
g = merge.groupby('nombre_area').agg({'postulaciones/avisos' : 'sum'}).sort_values(by
g.set_title('Top 10 areas con mas postulaciones por aviso', fontsize=20)
g.set_xlabel("Postulaciones/avisos", fontsize = 15, weight = 'bold')
g.set_ylabel("Area", fontsize = 15, weight = 'bold')
```

Out[37]: Text(0,0.5,'Area')




```
In [38]: g = merge.groupby('nombre_area').agg({'postulaciones/avisos' : 'sum'}).sort_values(by
         g.set_title('Top 10 areas con menor postulaciones por aviso', fontsize=20)
         g.set_xlabel("Postulaciones/avisos", fontsize = 15, weight = 'bold')
         g.set_ylabel("Area", fontsize = 15, weight = 'bold')
```

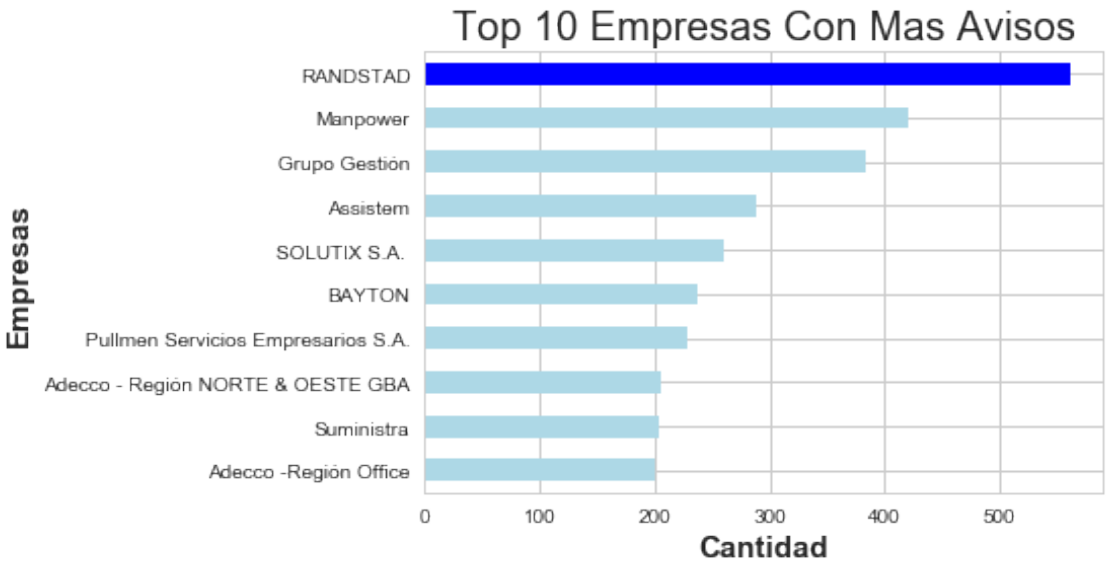
```
Out[38]: Text(0,0.5,'Area')
```

Conclusión: Programacion es el area que menor demanda tiene en relacion a su oferta, siendo que por cada aviso relacionado a programacion en promedio se postulan menos de 20 personas, en contraste con Recepcionista, a la cual por cada aviso se postulan en promedio mas de 800 personas. Tambien se observa que el mayor volumen de postulaciones se obtienen por trabajos no calificados, mientras que los que menos postulaciones tienen lo son (en mayor medida relacionados al rubro de la informatica).

6 £Alguna empresa esta acaparando mas postulaciones que las demás en relacion con su cantidad de avisos?

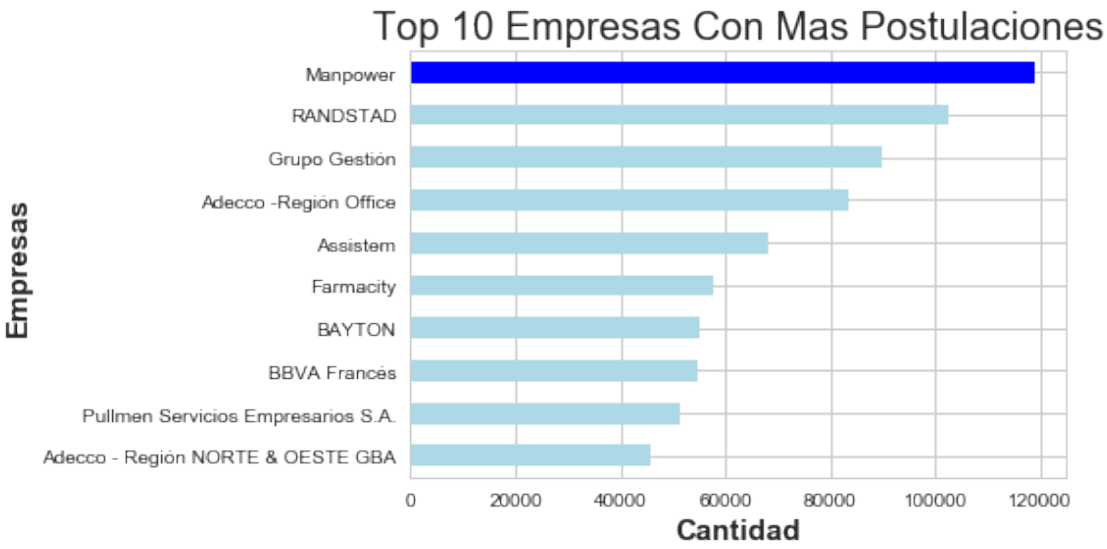
```
In [39]: empresas_con_mas_avisos = avisos_detalle_df['denominacion_empresa'].value_counts().hea
         empresas_con_mas_avisos.set_title("Top 10 Empresas Con Mas Avisos", fontsize = 20)
         empresas_con_mas_avisos.set_xlabel('Cantidad', fontsize = 15, weight = 'bold')
         empresas_con_mas_avisos.set_ylabel('Empresas', fontsize = 15, weight = 'bold')
```

```
Out[39]: Text(0,0.5,'Empresas')
```




```
In [40]: empresas_con_mas_postulaciones = merge_detalle_postulaciones_df['denominacion_empresa
         empresas_con_mas_postulaciones.set_title("Top 10 Empresas Con Mas Postulaciones", font
         empresas_con_mas_postulaciones.set_xlabel('Cantidad', fontsize = 15, weight = 'bold')
         empresas_con_mas_postulaciones.set_ylabel('Empresas', fontsize = 15, weight = 'bold')
```

```
Out[40]: Text(0,0.5,'Empresas')
```



Ahora vemos la relacion entre postulaciones y avisos pero con las empresas.

```
In [41]: #Teniendo en cuenta la ecuacion de Moivre, la desviacion estandard y el promedio de a
         merge = avisos_detalle_df['denominacion_empresa'].value_counts().reset_index()
         columnas = ['denominacion_empresa','cantidad_avisos']
         merge.columns = columnas
         print(merge['cantidad_avisos'].describe())
         merge = merge[ (merge['cantidad_avisos'] > 20 ) ]
         merge.head()

count    2591.000000
mean        5.219992
std        22.325433
min         1.000000
25%         1.000000
50%         1.000000
75%         3.000000
max       562.000000
Name: cantidad_avisos, dtype: float64


Out[41]:   denominacion_empresa  cantidad_avisos
         0             RANDSTAD              562
         1             Manpower              421
         2        Grupo Gestión              383
         3             Assistem              289
         4        SOLUTIX S.A.               260

In [42]: df = merge_detalle_postulaciones_df['denominacion_empresa'].value_counts().reset_index
         columnas = ['denominacion_empresa','cantidad_postulaciones']
```

```
df.columns = columnas
df.head()
```

Out[42]:

	denominacion_empresa	cantidad_postulaciones
0	Manpower	119013
1	RANDSTAD	102640
2	Grupo Gestión	89950
3	Adecco -Región Office	83530
4	Assistem	68125

In [43]:
```
merge = merge.merge(df, on = 'denominacion_empresa')
merge.head()
```

Out[43]:

	denominacion_empresa	cantidad_avisos	cantidad_postulaciones
0	RANDSTAD	562	102640
1	Manpower	421	119013
2	Grupo Gestión	383	89950
3	Assistem	289	68125
4	SOLUTIX S.A.	260	8206

In [44]:
```
#Obtenemos la relacion.
merge['postulaciones/avisos'] = ( merge['cantidad_postulaciones'] / merge['cantidad_av
merge.head()
```

Out[44]:

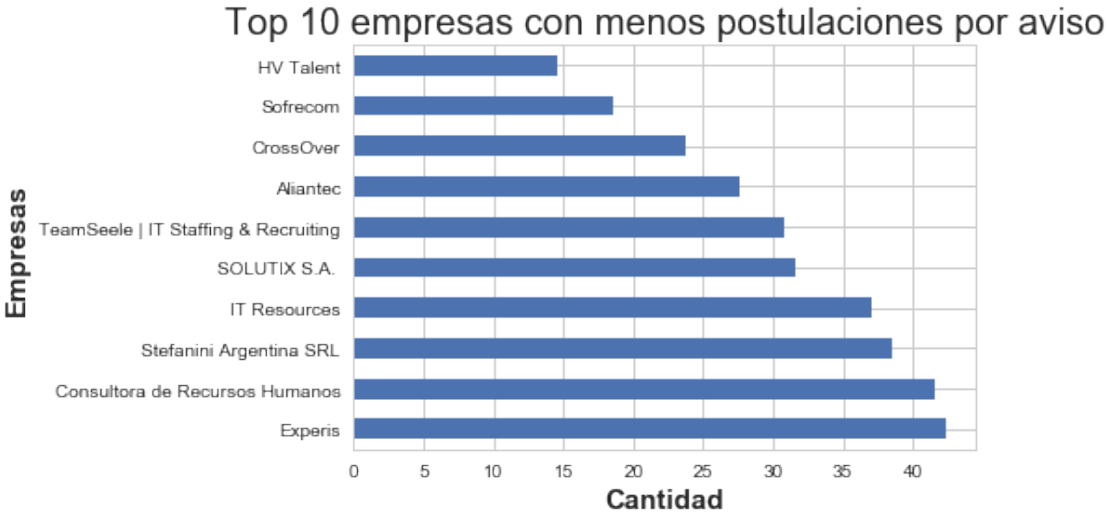
	denominacion_empresa	cantidad_avisos	cantidad_postulaciones	\
0	RANDSTAD	562	102640	
1	Manpower	421	119013	
2	Grupo Gestión	383	89950	
3	Assistem	289	68125	
4	SOLUTIX S.A.	260	8206	

	postulaciones/avisos
0	182.633452
1	282.691211
2	234.856397
3	235.726644
4	31.561538

In [45]:
```
#Grafico de relacion entre postulaciones y avisos.
plot = merge.groupby('denominacion_empresa').agg({'postulaciones/avisos' : 'sum'}).so
plot.set_title('Top 10 empresas con mas postulaciones por aviso', fontsize = 20)
plot.set_xlabel('Cantidad', fontsize = 15, weight = 'bold')
plot.set_ylabel('Empresas', fontsize = 15, weight = 'bold')
#Grafico de relacion entre postulaciones y avisos (por empresa)
```

Out[45]: Text(0,0.5,'Empresas')


```
In [46]: plot = merge.groupby('denominacion_empresa').agg({'postulaciones/avisos' : 'sum'}).so
         plot.set_title('Top 10 empresas con menos postulaciones por aviso', fontsize = 20)
         plot.set_xlabel('Cantidad', fontsize = 15, weight = 'bold')
         plot.set_ylabel('Empresas', fontsize = 15, weight = 'bold')
         #Grafico de relacion entre postulaciones y avisos (por empresa)

Out[46]: Text(0,0.5,'Empresas')
```



Nuevamente, las empresas relacionadas a informatica son las que menor cantidad de postulantes tienen.

7 £Cual es el rango de edad con mas postulaciones? £Cual es el promedio de edad de los postulantes para cada nivel laboral?

Para hallar la relacion entre nivel laboral y edad media de los postulantes se descartan los usuarios con fechas nulas o incorrectas.

```
In [47]: print('Cantidad original: 196138')
         print('')
         print('Fechas incorrectas :')
         print(postulantes_gye_df[postulantes_gye_df['fechanacimiento'].str.startswith('00')])
         print('')
         print('Cantidad de fechas incorrectas: ' , postulantes_gye_df[postulantes_gye_df['fecl
         df = postulantes_gye_df[ (postulantes_gye_df['fechanacimiento'].str.startswith('00'))
         print('')
         print('Cantidad sin fechas incorrectas : ', df.shape)
```

```
Cantidad original: 196138

Fechas incorrectas :
       idpostulante fechanacimiento  sexo
56206       xkPwXwY      0031-12-11   FEM
71458       LN85Y3b      0029-05-11  MASC
130846      8M2R6pz      0024-02-09   FEM
141832      A36Npjj      0033-09-14   FEM
148638      GNZOvAv      0004-07-19  MASC
149653      1QPQ8QL      0011-03-08  MASC

Cantidad de fechas incorrectas:  (6, 3)

Cantidad sin fechas incorrectas :  (195687, 3)
```

```
In [48]: #Ahora, sin las fechas incorrectas, puedo calcular la edad promedio.
         df.loc[:,'fechanacimiento'] = pd.to_datetime(df['fechanacimiento'])
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 195687 entries, 0 to 200886
Data columns (total 3 columns):
idpostulante       195687 non-null object
fechanacimiento    195687 non-null datetime64[ns]
sexo               195687 non-null object
dtypes: datetime64[ns](1), object(2)
memory usage: 6.0+ MB


C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:621: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item_labels[indexer[info_axis]]] = value
```

```
In [49]: #Calculo la edad de los postulantes y agrego la columna.
         df.loc[:,'edad'] = (2018 - df.fechanacimiento.dt.year)
         df.head()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:357: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[key] = _infer_fill_value(value)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:537: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
```

```
Out[49]:   idpostulante fechanacimiento  sexo  edad
         0         NM5M      1970-12-03   FEM    48
         1         5awk      1962-12-04   FEM    56
         2         ZaO5      1978-08-10   FEM    40
         3         NdJl      1969-05-09  MASC    49
         4         eo2p      1981-02-16  MASC    37
```

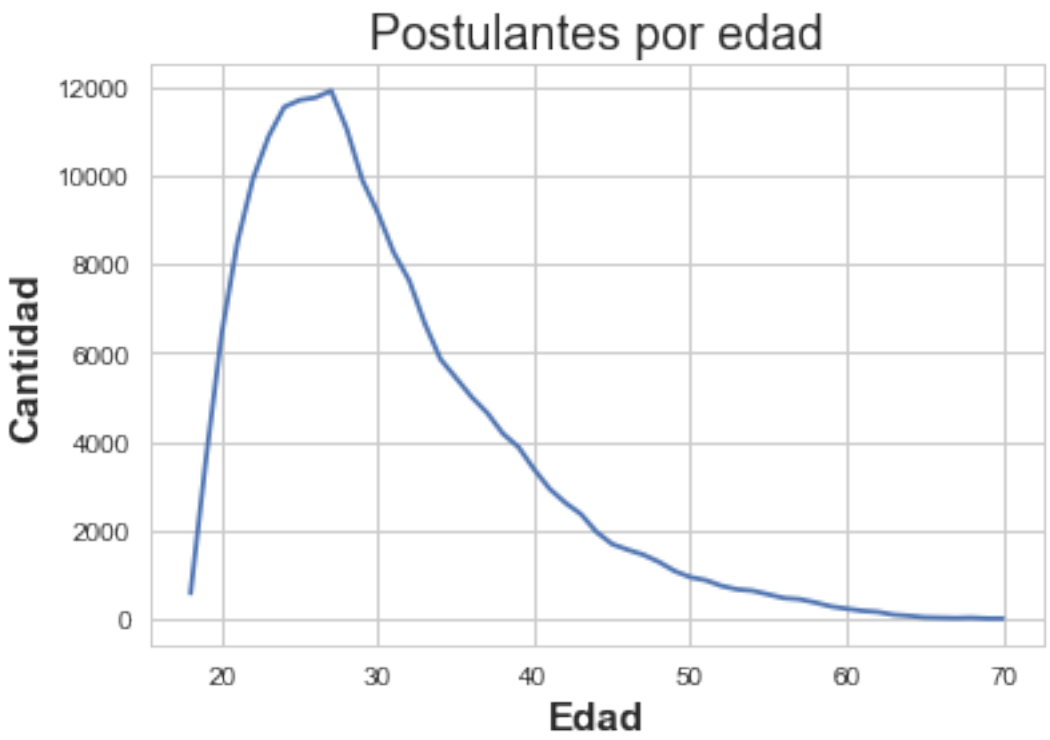
```
In [50]: #Hay postulantes con edades anormales, tambien los saco
         print('Postualantes con edad incorrecta: ')
         print(df[ ( df['edad'] > 70) | (df['edad'] < 18) ])

         df = df[ ( (df['edad'] > 70) | (df['edad'] < 18) ) == False ]
```

```
Postualantes con edad incorrecta:
       idpostulante fechanacimiento  sexo  edad
520          52DWRk      1946-12-03  MASC    72
6869         a5qWAm      2006-01-01  MASC    12
11671        epKzQ8      1944-03-23  MASC    74
15039        NqkN3L      1942-08-21   FEM    76
17969        ZBXbxP      1943-03-17  MASC    75
25080        8YZmej      1943-02-07  MASC    75
41689       A3AAAv5      1947-07-27  MASC    71
44744       GNJmLNv      1946-06-16  MASC    72
49323       JBmkewO      1947-10-13  MASC    71
52616       EzpKa56      1947-06-30   FEM    71
```

```
55962       ZDaVBez      1942-07-16  MASC    76
63142       Pma4rYv      1947-01-21   FEM    71
64272       bO4WQVY      1944-08-02  MASC    74
83825       xkppYAY      1939-07-27  MASC    79
97800       8ML48lz      1947-07-12  MASC    71
98648       bOAkpEq      1944-11-13  MASC    74
99345       6rLd8RL      1941-09-21  MASC    77
115060      VNP3DZE      1947-12-04  MASC    71
145374      EzeD402      1947-11-27   FEM    71
154559      xkdvwm0      1775-07-09  MASC   243
156483      4rP8l0R      1921-08-18   FEM    97
164618      96X1loa      1917-07-08  MASC   101
199706      X95Jowv      1942-12-19  MASC    76
```

In [51]:
```
#Visualizacion de las postulaciones segun la edad del postulante.
g = df.edad.value_counts().sort_index().plot()
g.set_title('Postulantes por edad', fontsize=18)
g.set_xlabel('Edad', fontsize = 15, weight = 'bold')
g.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')
```

Out[51]:
```
Text(0,0.5,'Cantidad')
```

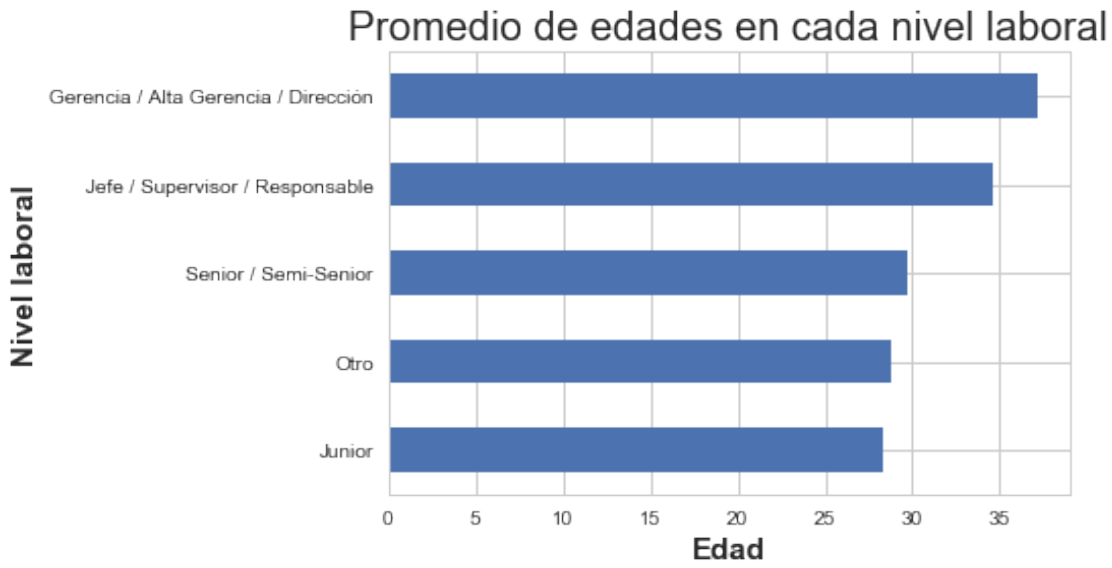


In [52]:
```
#Ahora junto la informacion de las edades con el data frame de postulaciones-avisos.
merge = merge_detalle_postulaciones_df.merge(df, on = 'idpostulante')
merge.head()
print(merge.shape)
```

```
(3028479, 14)
```

In [53]:
```
#Relaizamos un grafico de barras para mostrar la edad promedio de cada nivel laboral.
g = merge.groupby('nivel_laboral').agg({'edad':'mean'}).sort_values(by = 'edad').plot
g.set_title('Promedio de edades en cada nivel laboral', fontsize = 20)
g.set_xlabel('Edad', fontsize = 15, weight = 'bold')
g.set_ylabel('Nivel laboral', fontsize = 15, weight = 'bold')
```

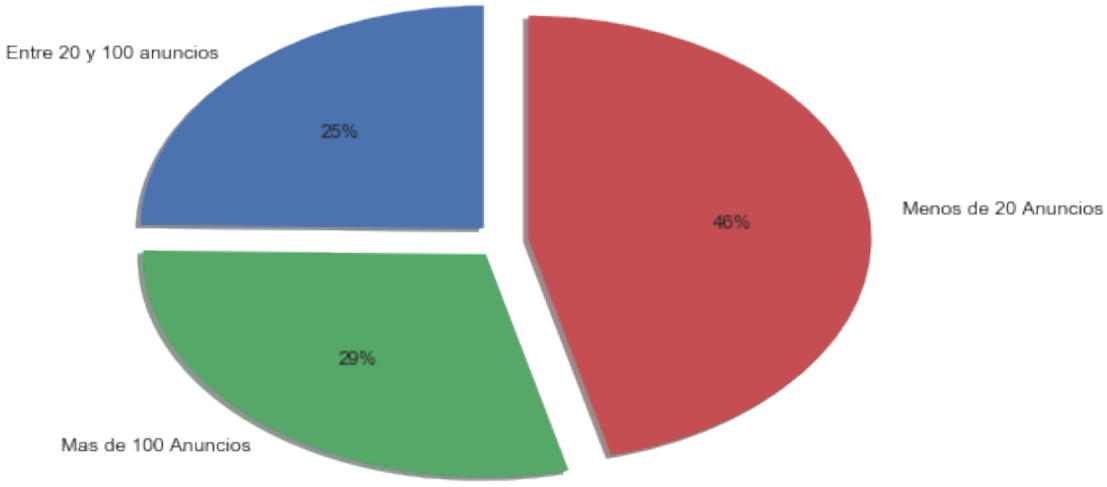
Out[53]: Text(0,0.5,'Nivel laboral')



In [54]:
```
#ANUNCIOS POR EMPRESA
empresas_con_mas_avisos = avisos_detalle_df['denominacion_empresa'].value_counts().to

menos_de_20 = empresas_con_mas_avisos[empresas_con_mas_avisos['denominacion_empresa']
menos_de_20 = menos_de_20['denominacion_empresa'].sum()

mas_de_100 = empresas_con_mas_avisos[empresas_con_mas_avisos['denominacion_empresa'] >
mas_de_100 = mas_de_100['denominacion_empresa'].sum()

otros = empresas_con_mas_avisos[(empresas_con_mas_avisos['denominacion_empresa'] > 20)
otros = otros['denominacion_empresa'].sum()
```

8 Anuncios por empresa

In [55]:
```
pie_chart = plt.pie([otros,mas_de_100,menos_de_20], labels = ['Entre 20 y 100 anuncios
```



In [56]:
```
#Trabajar con las postulaciones unicamente.
```

In [57]:
```
postulaciones_df['fechapostulacion'] = pd.to_datetime(postulaciones_df['fechapostulaci
plot = postulaciones_df['fechapostulacion'].dt.hour.value_counts().sort_index().plot(
plot.set_title('Postulaciones segun hora del dia', fontsize = 20)
plot.set_xlabel('Hora del dia', fontsize = 15, weight = 'bold')
plot.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')
```

Out[57]:
```
Text(0,0.5,'Cantidad')
```

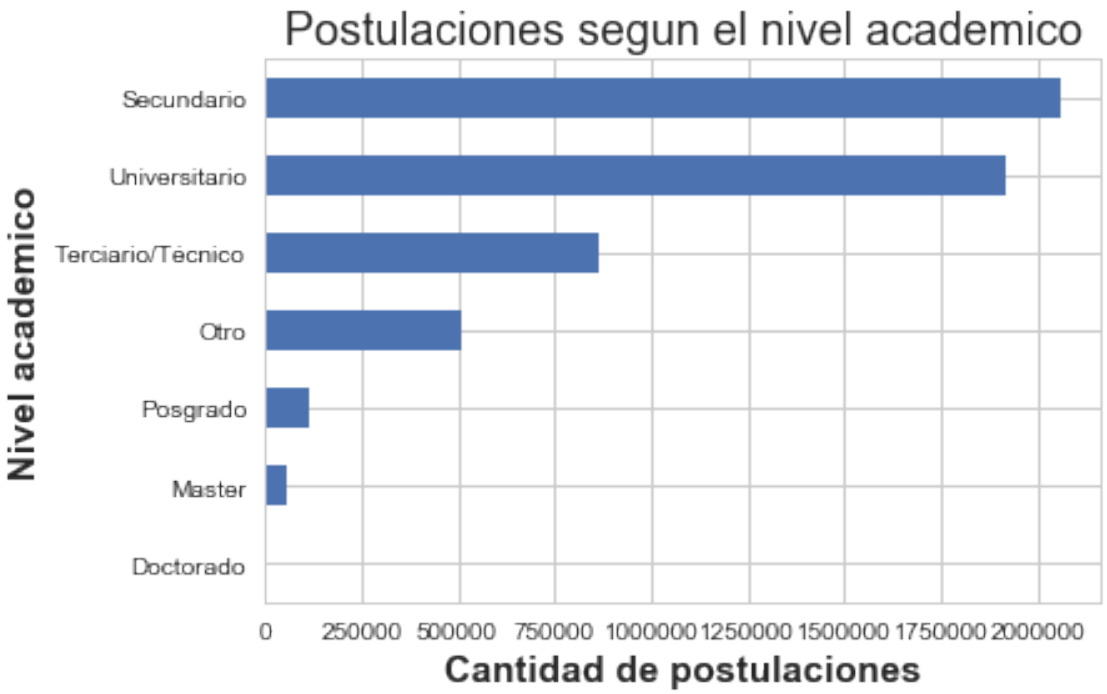
Viendo el grafico se puede notar que el rango horario mas activo en cuanto a postulaciones es la media mañana. El flujo de gente que ingresa en este horario al sitio es considerable.

```
In [58]: df = postulantes_edu_df.merge(postulaciones_df, on='idpostulante')
         df['cantidad'] = 1
         df.groupby('nombre').agg({'cantidad':'count'}).sort_values(by = 'cantidad')
```

Out[58]:

nombre	cantidad
Doctorado	3552
Master	58514
Posgrado	116924
Otro	508217
Terciario/Técnico	864464
Universitario	1916208
Secundario	2060789

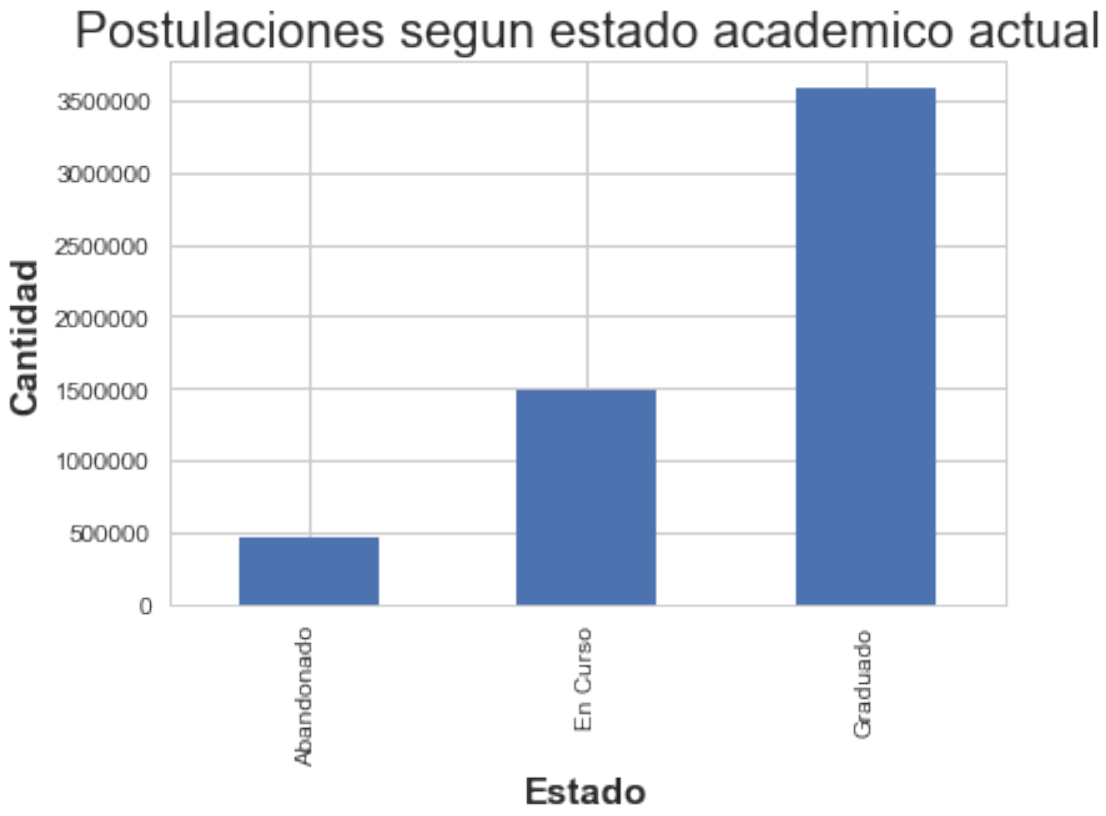
```
In [59]: plot = df.groupby('nombre').agg({'cantidad':'count'}).sort_values(by = 'cantidad').plo
         plot.set_title('Postulaciones segun el nivel academico', fontsize = 18)
         plot.set_xlabel('Cantidad de postulaciones', fontsize = 15, weight = 'bold')
         plot.set_ylabel('Nivel academico', fontsize = 15, weight = 'bold')

Out[59]: Text(0,0.5,'Nivel academico')
```



In [60]:
```
#Flitrar nivel academico por estado completado.
```

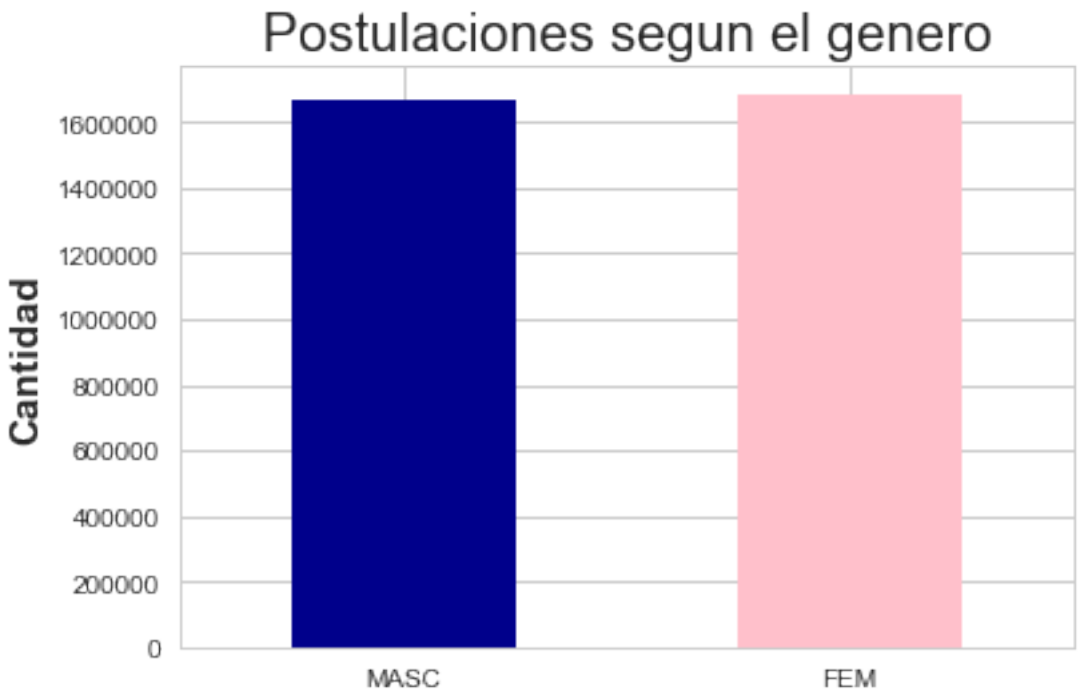
In [61]:
```
plot = df.groupby('estado').agg({'cantidad':'count'}).plot(kind = 'bar', legend = Fals
plot.set_title('Postulaciones segun estado academico actual', fontsize = 20)
plot.set_xlabel('Estado', fontsize = 15, weight = 'bold')
plot.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')
```

Out[61]:
```
Text(0,0.5,'Cantidad')
```



El mayor flujo de postulaciones lo dan los postulantes graduados.

```
In [62]: df = postulantes_gye_df.merge(postulaciones_df, on='idpostulante')
         plot = df['sexo'].value_counts().sort_values().plot('bar', color = ['darkblue','pink']
         plot.set_title('Postulaciones segun el genero', fontsize = 20)
         plot.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')

Out[62]: Text(0,0.5,'Cantidad')
```



9 Ahora queremos ver la relaciones entre las vistas y las postulaciones.

```
In [63]: #vistas_df['timestamp'] = pd.to_datetime(vistas_df['timestamp'])
         postulaciones_df['fechapostulacion'] = pd.to_datetime(postulaciones_df['fechapostulaci

In [64]: postulaciones_df['fechapostulacion'].dt.month.value_counts()

Out[64]: 2    2125425
         1    1276198
         Name: fechapostulacion, dtype: int64

In [65]: vistas_df['timestamp'].dt.month.value_counts()

Out[65]: 2    921074
         3     40823
         Name: timestamp, dtype: int64
```

Observamos que solo hay datos en comun en el mes de febrero.

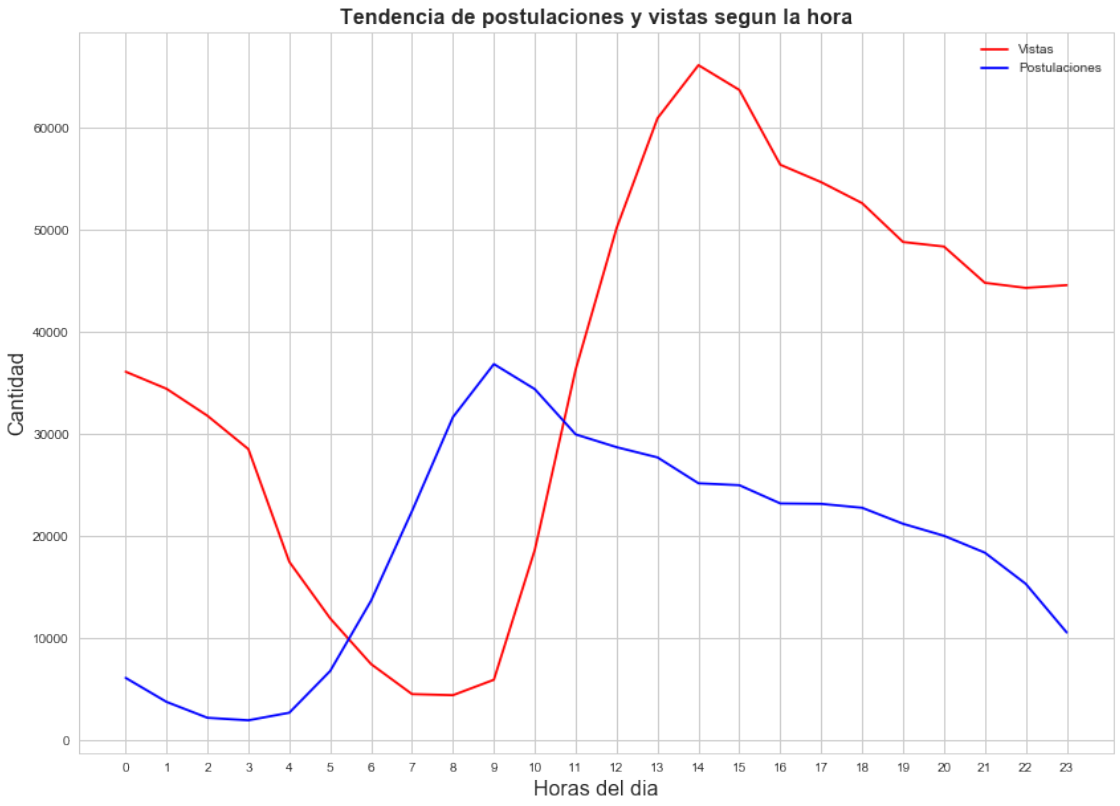
```
In [66]: #Observamos que la informacion disponible es de solo la ultima semana de febrero, por
         #Filtramos los dias que tienen datos incompletos (viernes 23 de febrero y jueves 1 de
         postulaciones_filtradas = postulaciones_df[(postulaciones_df['fechapostulacion'].dt.mo
         vistas = vistas_df[(vistas_df['timestamp'].dt.month == 2 ) & ((vistas_df['timestamp']
         postulaciones_filtradas.head()
```

Out[66]:

	idaviso	idpostulante	fechapostulacion
12	1112315188	5awk	2018-02-28 15:54:28
13	1112346738	5awk	2018-02-28 15:54:47
14	1112330625	5awk	2018-02-28 15:56:43
15	1112306543	5awk	2018-02-28 15:57:35
16	1112315170	5awk	2018-02-28 15:59:14

In [67]:
```
#Falta hacer un label de las lineas

fig = plt.figure() #Creates a figure for the next plot, then disapears

ax = fig.add_subplot(111)#Form and background

plt.title('Tendencia de postulaciones y vistas segun la hora',fontsize=16,fontweight=

ax.set_xlabel('Horas del dia',fontsize=16)


vistas['timestamp'].dt.hour.value_counts().sort_index().plot(color='red', label = 'Vis
postulaciones_filtradas['fechapostulacion'].dt.hour.value_counts().sort_index().plot(
ax.set_ylabel('Cantidad',fontsize=16,)
```

Out[67]: Text(0,0.5,'Cantidad')



En este grafico notamos una irregularidad entre las 5 y las 10 horas. Suponemos que la gran diferencia de vistas y postulaciones se debe a la forma en la que se recopilan los datos. Por ejemplo, puede ser que a la hora de postularse en un aviso, no se cuente la vista al mismo.

```
In [68]: #Ahora empezamos a trabajar con los datos para analizar por día.
         vistas  = vistas['timestamp'].dt.weekday_name.to_frame()
         vistas['timestamp'] = pd.Categorical(vistas['timestamp'], categories=['Monday','Tuesda
         postulaciones_filtradas = postulaciones_filtradas['fechapostulacion'].dt.weekday_name
         postulaciones_filtradas['fechapostulacion'] =pd.Categorical(postulaciones_filtradas['
```

```
In [69]: print(postulaciones_filtradas['fechapostulacion'].value_counts().sort_index())
         print()
         print(vistas['timestamp'].value_counts().sort_index())
```

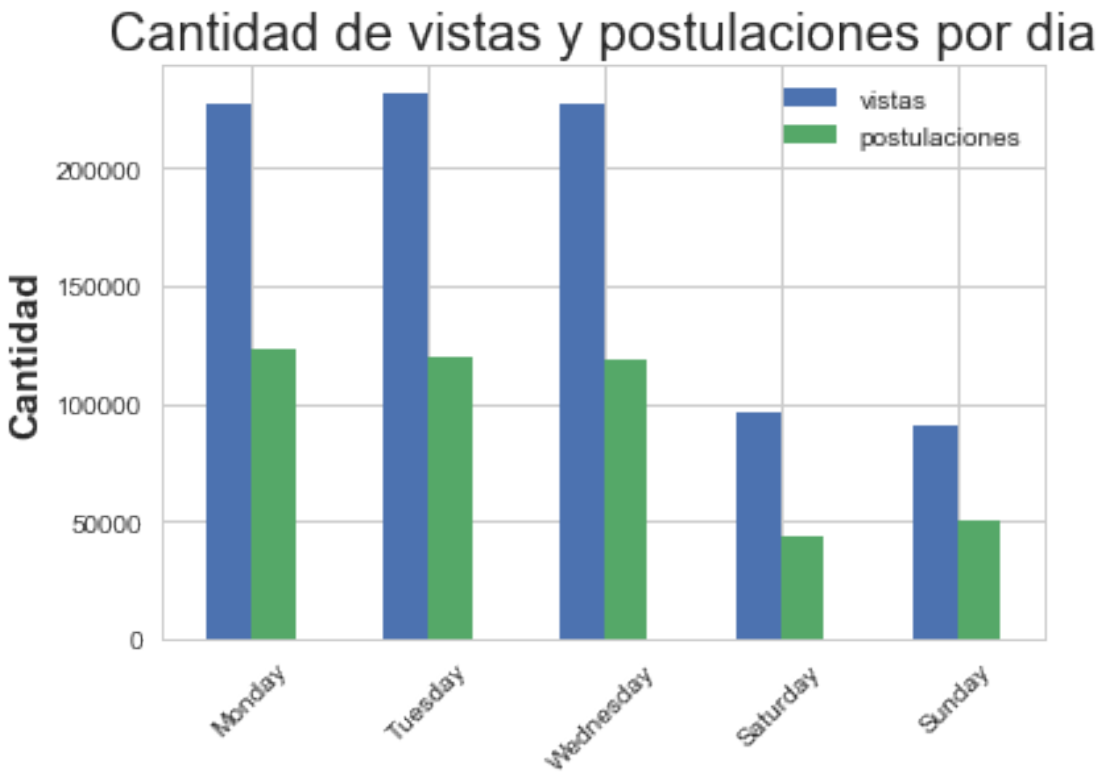
```
Monday       122688
Tuesday      119845
Wednesday    118296
Saturday      43718
Sunday        49924
Name: fechapostulacion, dtype: int64

Monday       227957
Tuesday      232145
Wednesday    227160
Saturday      95930
Sunday        90646
Name: timestamp, dtype: int64
```

```
In [70]: df = vistas['timestamp'].value_counts().to_frame()
         df['postulaciones'] = postulaciones_filtradas['fechapostulacion'].value_counts()
         df.columns = ['vistas','postulaciones']

         df.sort_index(inplace=True)
         df.head()
```

```
Out[70]:            vistas  postulaciones
         Monday     227957         122688
         Tuesday    232145         119845
         Wednesday  227160         118296
         Saturday    95930          43718
         Sunday      90646          49924
```

```
In [71]: g = df.plot(kind='bar', rot = 45)
         g.set_title('Cantidad de vistas y postulaciones por dia', fontsize = 20)
         g.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')
```

```
Out[71]: Text(0,0.5,'Cantidad')
```



Viendo solo una cantidad muy limitada de datos de algunos dias de febrero, podemos observar que la cantidad tanto de vistas como postulaciones disminuye considerablemente el fin de semana.

```
In [72]: #vistas promedio por persona en el rango dado

         vistas_df['cantidad'] = 1
         vistas_per_capita = vistas_df.groupby('idpostulante').agg({'cantidad':'count'})
         vistas_per_capita['cantidad'].mean()
```

```
Out[72]: 11.000274464508308
```

10 £Las palabras en los títulos de los avisos pueden predecir o atraer postulaciones?

```
In [73]: from collections import Counter
         #Hacemos una lista con todas las palabras que aparecen en los titulos y sus aparicion
         contador_palabras_avisos = Counter(" ".join(avisos_detalle_df['titulo'].values.tolist

         #Formo un DF con esa lista

         contador_palabras_avisos = list(contador_palabras_avisos)
```

```
contador_palabras_avisos_df = pd.DataFrame(contador_palabras_avisos)
contador_palabras_avisos_df.head()

contador_palabras_avisos_df.columns=['palabra','apariciones']
contador_palabras_avisos_df=contador_palabras_avisos_df.set_index('palabra')

#Saco los adverbios del df
adverbios=['','-','de','con','en','y','a','para','la','al','los','/']
contador_palabras_avisos_df=contador_palabras_avisos_df.drop(adverbios)
```

In [74]:
```
plot = contador_palabras_avisos_df.sort_values(by='apariciones',ascending=False).head
plot.set_title('Palabras mas frecuentes en titulos de avisos', fontsize = 20)
plot.set_xlabel('Apariciones', fontsize = 15, weight = 'bold')
plot.set_ylabel('Palabras', fontsize = 15, weight = 'bold')
```

Out[74]: Text(0,0.5,'Palabras')

In [75]:
```
#Por cada postulacion contamos las palabras que aparecen en el titulo del aviso al qu
contador_palabras_postulaciones = Counter(" ".join(merge_detalle_postulaciones_df['ti

contador_palabras_postulaciones = list(contador_palabras_postulaciones)
```

```
contador_palabras_postulaciones_df=pd.DataFrame(contador_palabras_postulaciones)

contador_palabras_postulaciones_df.columns=['palabra','apariciones']
contador_palabras_postulaciones_df=contador_palabras_postulaciones_df.set_index('palal

contador_palabras_postulaciones_df=contador_palabras_postulaciones_df.drop(adverbios)
```

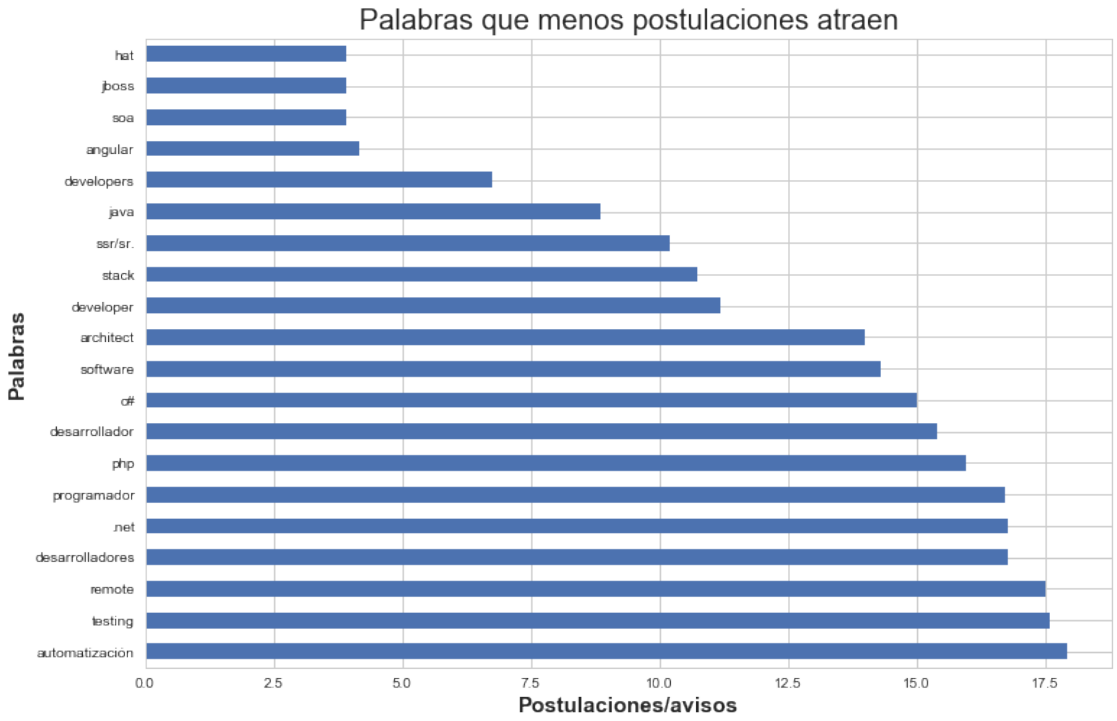
In [76]:
```
plot = contador_palabras_postulaciones_df.sort_values(by='apariciones',ascending=Falsc
plot.set_title('Palabras mas frecuentes en titulos de avisos a los que se postularon'
plot.set_xlabel('Apariciones', fontsize = 15, weight = 'bold')
plot.set_ylabel('Palabras', fontsize = 15, weight = 'bold')
```

Out[76]: Text(0,0.5,'Palabras')

In [77]:
```
# Nos quedamos con las palabras que figuran en al menos 20 avisos
contador_palabras_avisos_df_filtrado=contador_palabras_avisos_df.loc[contador_palabras

efectividad_palabras_df = contador_palabras_postulaciones_df.sort_values(by='aparicion
efectividad_palabras_df = efectividad_palabras_df.dropna()
efectividad_palabras_df.columns=['postulaciones/avisos']
```

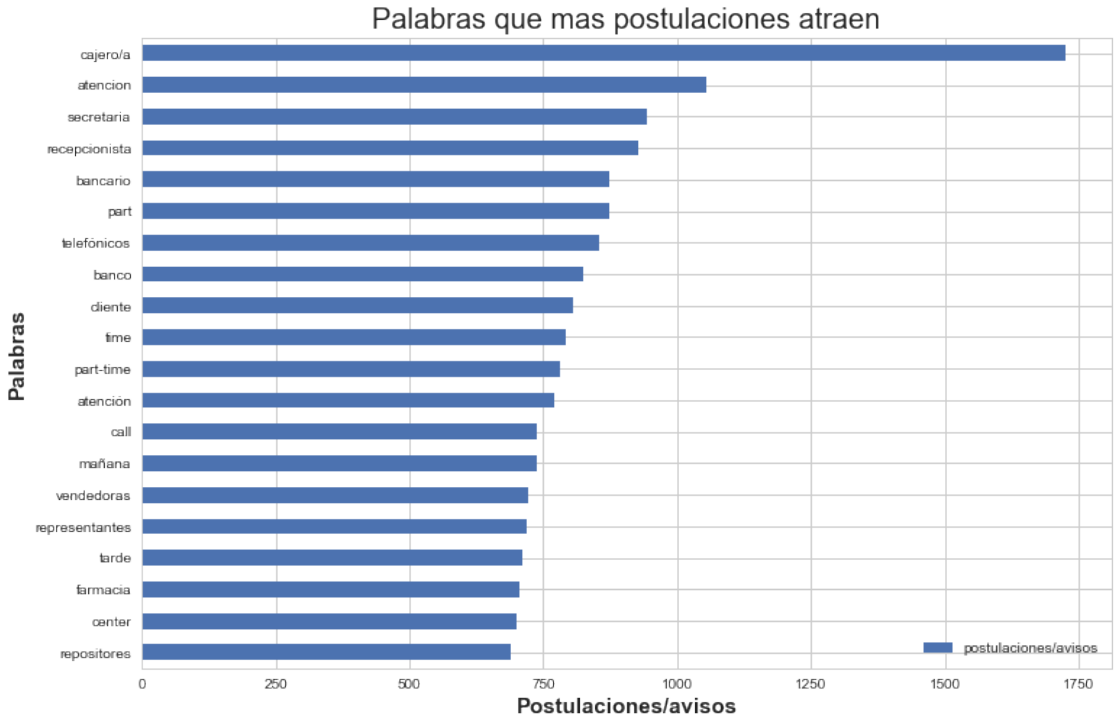
In [78]:
```
plot = efectividad_palabras_df.sort_values(by='postulaciones/avisos',ascending= False)
plot.set_title('Palabras que menos postulaciones atraen', fontsize = 20)
plot.set_xlabel('Postulaciones/avisos', fontsize = 15, weight = 'bold')
plot.set_ylabel('Palabras', fontsize = 15, weight = 'bold')
```

Out[78]: Text(0,0.5,'Palabras')



In [79]:
```
plot = efectividad_palabras_df.sort_values(by='postulaciones/avisos',ascending= False)
plot.set_title('Palabras que mas postulaciones atraen', fontsize = 20)
plot.set_xlabel('Postulaciones/avisos', fontsize = 15, weight = 'bold')
plot.set_ylabel('Palabras', fontsize = 15, weight = 'bold')
```

Out[79]: Text(0,0.5,'Palabras')



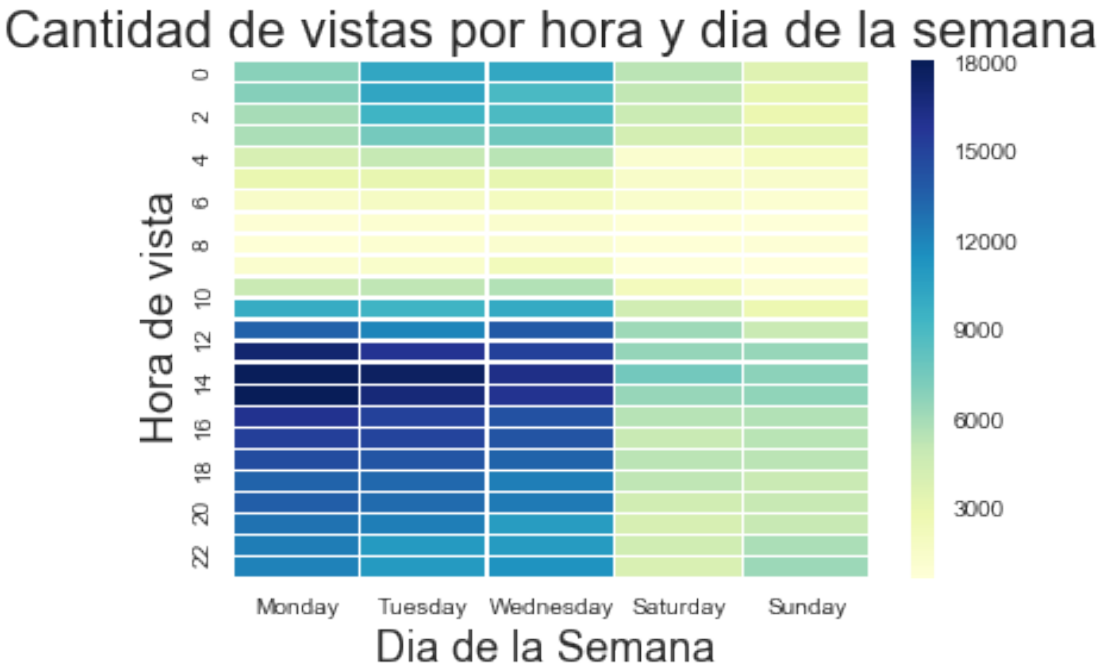
Algunas conclusiones: Puede verse que el empleo de cajero tiene significativamente más postulaciones por aviso que los demás, lo que explica el por qué el area de ventas figura en la primer posicion del top de areas con mas postulaciones. Tambien puede verse que los anuncios con palabras relacionadas a programación tienen relativamente muy pocas postulaciones (£Algo bueno para nosotros?). Ademas, si bien analista es la palabra que mas aparece en los titulos de las postulaciones, esto seguramente se deba a que es la que mas figura en todos los avisos en general.

11 Mas graficos con informacion de postulaciones y vistas.

Comenzamos a trabajar los datos para generar un heatmap. Primero mostrando las vistas segun el horario y el dia de la semana. Luego las postulaciones segun el horario y dia de la semana.

```
In [80]: vistas_df.rename(columns={'idAviso': 'idaviso'}, inplace= True)
         df= avisos_detalle_df.merge(vistas_df, on= 'idaviso')
         df['timestamp'] =pd.to_datetime(df['timestamp'])
         df['Weekday'] = df['timestamp'].map(lambda x:x.weekday_name)
         df['Hour']=df['timestamp'].dt.hour
         df['count']= 1

In [81]: for_heatmap = df.pivot_table(index='Hour', columns='Weekday', values='count', aggfunc=
         for_heatmap = for_heatmap.reindex(['Monday','Tuesday','Wednesday','Saturday','Sunday'

In [82]: g = sns.heatmap(for_heatmap,linewidths=.5,  cmap="YlGnBu")
         g.set_title("Cantidad de vistas por hora y dia de la semana", fontsize=22)
         g.set_xlabel("Dia de la Semana ",fontsize=18)
         g.set_ylabel(" Hora de vista", fontsize=18)
```

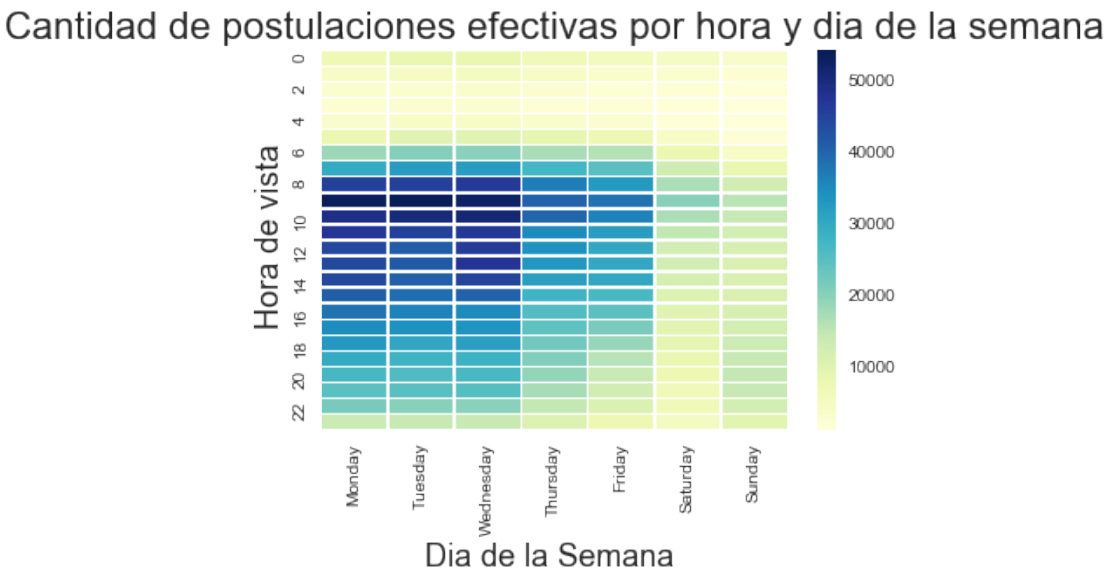
Out[82]: Text(34,0.5,' Hora de vista')



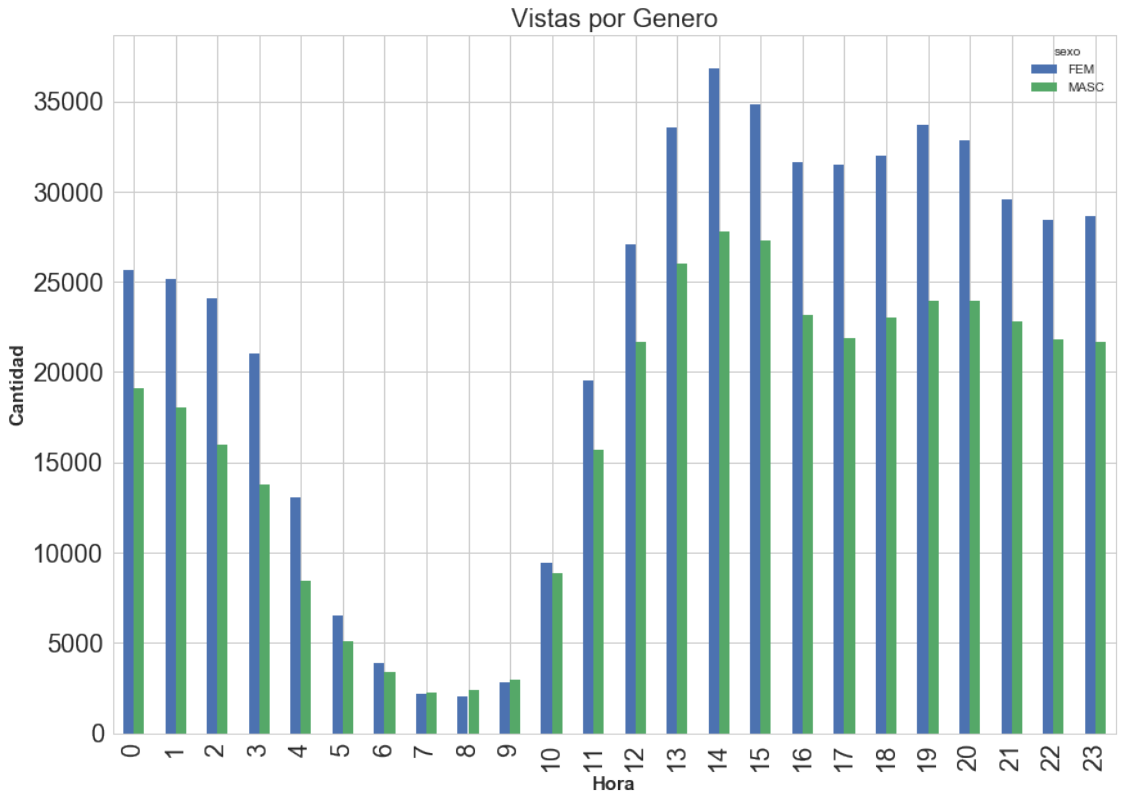
In [83]:
```
df = postulaciones_df['fechapostulacion'].to_frame()

df['fechapostulacion'] =pd.to_datetime(df['fechapostulacion'])
df['Weekday'] = df['fechapostulacion'].map(lambda x:x.weekday_name)
df['Hour']=df['fechapostulacion'].dt.hour
df['count']= 1
```

In [84]:
```
for_heatmap = df.pivot_table(index='Hour', columns='Weekday', values='count', aggfunc=
for_heatmap = for_heatmap.reindex(['Monday','Tuesday','Wednesday','Thursday','Friday'
g = sns.heatmap(for_heatmap,linewidths=.5,  cmap="YlGnBu")
g.set_title("Cantidad de postulaciones efectivas por hora y dia de la semana", fontsi
g.set_xlabel("Dia de la Semana ",fontsize=18)
g.set_ylabel(" Hora de vista", fontsize=18)
```

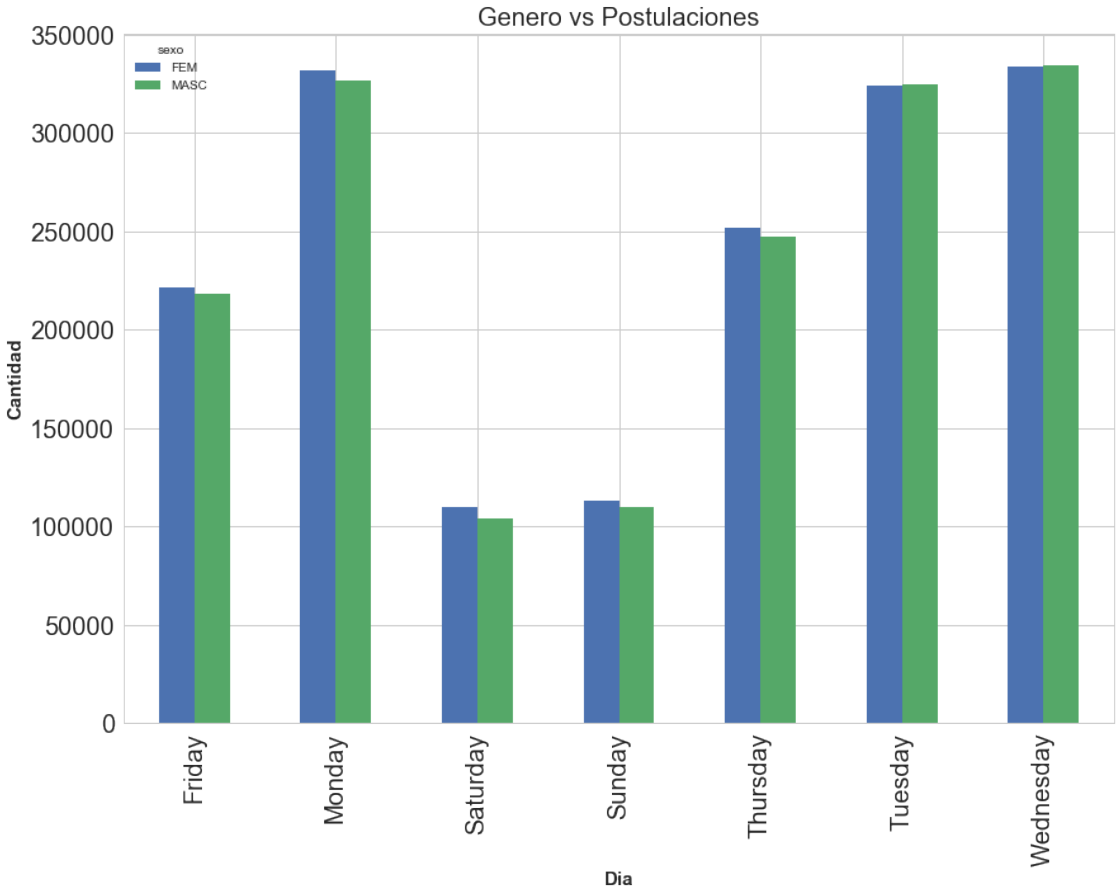
Out[84]: Text(34,0.5,' Hora de vista')



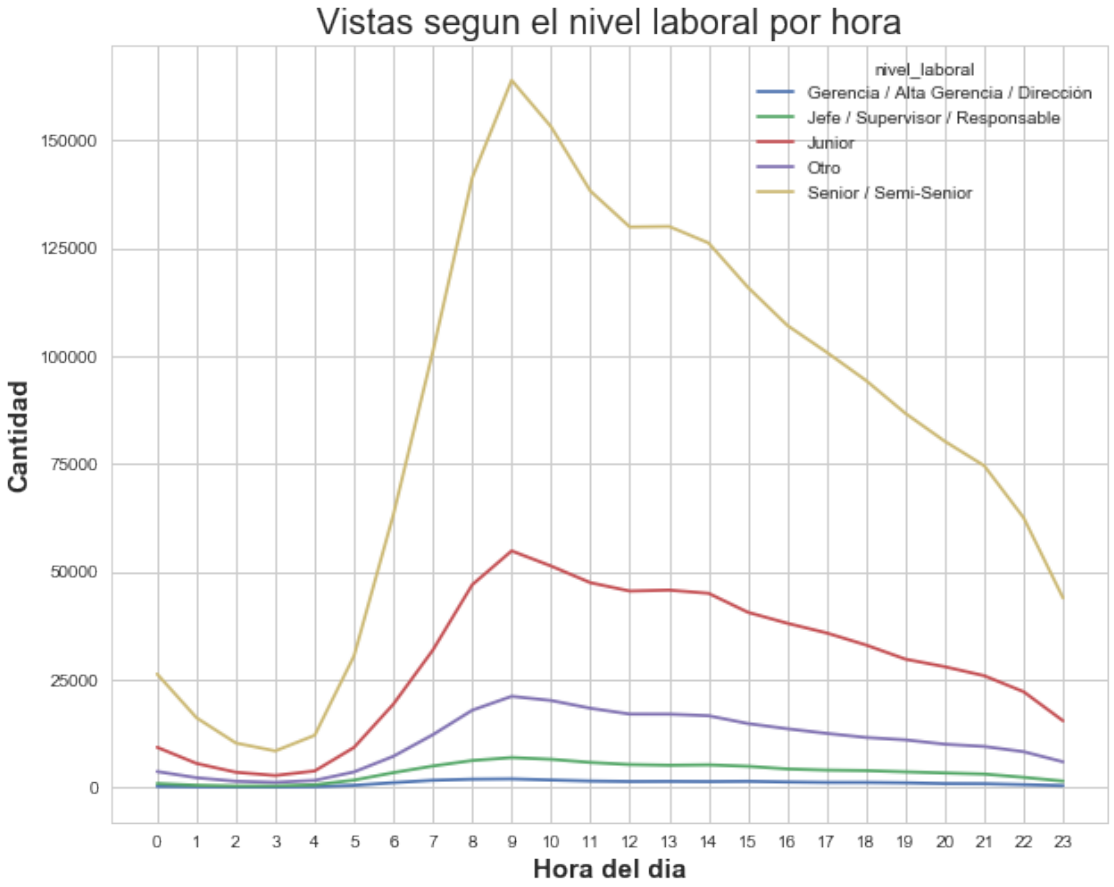

```
In [85]: df = vistas_df.merge(postulantes_gye_df, on='idpostulante')
         df['timestamp'] = pd.to_datetime(df['timestamp'] )
         df['timestamp'] = df['timestamp'].dt.hour
         plot = pd.crosstab(df.timestamp,df.sexo).plot(kind='bar',figsize=(14,10),fontsize=20)
         plot.set_title('Vistas por Genero',fontsize=20)
         plot.set_xlabel('Hora', fontsize = 15, weight = 'bold')
         plot.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')

Out[85]: Text(0,0.5,'Cantidad')
```




```
In [86]: df = postulaciones_df.merge(postulantes_gye_df, on='idpostulante')
         df['fechapostulacion'] = pd.to_datetime(df['fechapostulacion'] )
         df['fechapostulacion'] = df['fechapostulacion'].dt.weekday_name
         plot = pd.crosstab(df['fechapostulacion'],df['sexo']).plot(kind='bar',figsize=(14,10)
         plot.set_title('Genero vs Postulaciones',fontsize=20)
         plot.set_xlabel('Dia', fontsize = 15, weight = 'bold')
         plot.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')

Out[86]: Text(0,0.5,'Cantidad')
```




```
In [87]: df = postulaciones_df.merge(avisos_detalle_df, on='idaviso')
         df['fechapostulacion'] =pd.to_datetime(df['fechapostulacion'])
         df['fechapostulacion'] =df['fechapostulacion'].dt.hour
         plot = pd.crosstab(df['fechapostulacion'],df['nivel_laboral']).plot(kind='line',figsi
         plot.set_title('Vistas segun el nivel laboral por hora', fontsize = 20)
         plot.set_xlabel('Hora del dia', fontsize = 15, weight = 'bold')
         plot.set_ylabel('Cantidad', fontsize = 15, weight = 'bold')

Out[87]: Text(0,0.5,'Cantidad')
```


Vistas segun el nivel laboral por hora
nivel_laboral
Gerencia / Alta Gerencia / Dirección
Jefe / Supervisor / Responsable
Junior
Otro
Senior / Semi-Senior
Cantidad
150000
125000
100000
75000
50000
25000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hora del dia